MemCom

An Integrated Memory and Data Management System

MemCom Monitor User Manual

SMR *Engineering & Development*

MemCom

An Integrated Memory and Data Management System

MemCom Monitor User Manual

SMR *Engineering & Development*

MemCom Monitor User Manual, Version 6.6, patch 5 (November 2001)

SMR Corporation
P. O. Box 4014
CH-2500 Bienne 4
Switzerland

Tel: +41-32-3452121
Fax: +41-32-3452120
Email: support@smr.ch

Contents

1 Introduction

The *MemCom Monitor* is a command language driven program which offers most of the *MemCom* data base manager (*DB*) and table manager (*TB*) library functions in an interactive environment. It provides means for

- Inquiring the contents of *MemCom* data files
- Modifying, merging and altering the contents of *MemCom* data files
- Performing neutral data file operations

The *Monitor* can easily be used to check data while application programs are running, as well as for pre- and post-conditioning of data. As an example, during Finite Element computations it is often desirable to read, change or move specific information stored on data base files. This can easily be done using the *Monitor*. The *Monitor* may also be used as an instrument for checking and debugging of codes.

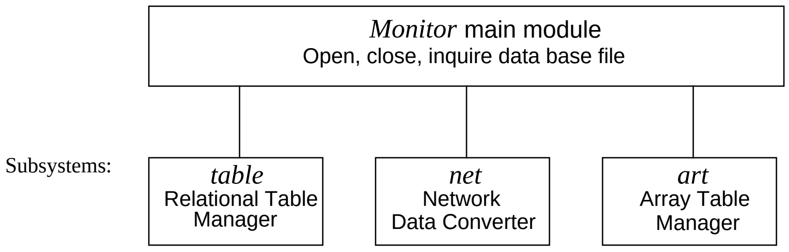
The *Monitor* is organized in subsystems similar to the *MemCom* library. The command module provides the interface to access objects on *MemCom* data base files. The following *Monitor* subsystems are presently supported:

- *table* (Relational table manager)
- *net* (Neutral file manager)
- *art* (Array table manager)

In Chapter 2 the *Monitor* commands are explained and in Chapter 3 some hints on how to use the *MemCom Monitor* are given.

2 The Monitor Command Language

The *Monitor* syntax rules and the *Monitor* commands are described in this Chapter. *Monitor* commands are logically grouped into sets of modules:



Each module prints a prompt string when ready to accept input. The *Monitor* command module prompts by

```
MON-DB->
```

You may then enter *Monitor* commands according to the syntax rules below. Note that *Monitor* commands are not case-sensitive. However, data set names and table keywords are case-sensitive. *Monitor* commands may logically be separated into the following groups of commands:

- Data base file management operations.
- Data set manipulations.
- Data set descriptor and relational table manipulations, *see table subsystem*.
- Array table manipulations, see art subsystem.
- Neutral file data converter, see net subsystem.
- Utility commands, see *utilities*.

Subsystems contain groups of *Monitor* commands within an independent module. The table displayed above summarizes the subsystems presently available. Subsystems are quit by typing *end*.

The data base manager module operates in two modes:

- ***DB*** Data base manager mode (default)
  All data manipulations take place directly on the data base files. The contents of the data base files are immediately changed
- ***AM*** Access manager mode
  All data is manipulated through the Dynamic Common (DC). Data base files are not changed until *save*, *close* or *load* commands are issued. Although the DC is always associated to a data file (if such a file has been specified) no data is exchanged with the data base files if this is not explicitly required.

Observe that some commands do not hold for both the *AM* and *DB* modes.

The following paragraphs describe all *Monitor* commands. Appendix A explains the *Monitor start-up procedure*. Appendix B contains a summary of the *MemCom data types*.

Syntax rules

The *Monitor* command language consists of a keyword-driven language, i.e. instructions are in the form of

key **attribute(s)** key **attribute(s)** etc.

The command dictionary checks if the keywords are part of the command language grammar. If a matching keyword cannot be found in the application dictionary, the command language dictionary is scanned. Thus, the statement

```
print COOR.*
```

is recognized by the *Monitor* application dictionary, whereas the statement

```
while (cond) (body)
```

is processed by the command language interpreter. The command language interpreter essentially discerns between *Command Keywords*, *Expressions* (both arithmetic and logical), *Flow control* commands and *Procedures*. All input to the interpreter is processed on the fly, i.e. statement by statement, except for flow control statements and expressions, i.e. statement placed between ellipses. Statements placed between ellipses are evaluated once matching ellipses are found.

Since the command language is based on pre-fixed notation, all instructions must be terminated by a separator symbol. The semi-colon “;” separates instructions. When in interactive mode the carriage-return carries out the same task, except where instructions are placed in flow control statements, i.e. *loop* and *if* bodies.

Keywords are printed in capitals throughout this document although upper and lower case letters may be mixed arbitrarily, i.e. keys are not case-sensitive. Keywords must start by *A-Z* or *a-z* and must be delimited by blanks or semicolons (if there are no attributes). Keywords may only contain the characters *a-z A-Z 0-9 _*. You may abbreviate keys. The minimum number of characters required depends on the uniqueness of the keyword within the current context.

Attributes consist of integer numbers, integer number lists, real numbers (floating point numbers), real number lists, character strings and text.

Integer numbers are represented by strings which contain the characters *0-9 + -* (examples: 1000, -123, +1).

Real numbers are represented by strings which contain the characters *0-9 + - . d D e E* (examples: -1.2345e-7, +.00002, 10).

Character strings may consist of any combination of characters except *blank @ [ ] ! ;*.

Text strings may contain any characters and must be delimited by quotes or any by other character not contained in the string.

Lists are collections of real or integer data. Whenever lists are to be specified, they must be enclosed by brackets:

```
[ 1 10:20 45 46 50 ]
```

generates a list of integers in the form of

```
[ 1 10 11 12 13 14 15 16 17 18 19 20 45 46 50 ]
```

Thus, expressions of the form

```
from:to:step
```

within lists generate sequences of integer or real data.

Comments, i.e. explanatory text, may be placed in scripts. Comments must start with the # character.

The following special symbols are used in the command description sections:

- Ellipses in the specification section mean that the parameters between ellipses are optional.
- The vertical bar | means “or” i.e. one of the parameters listed may be selected.

Expressions

Expressions, whether arithmetic or logical, must be placed between ellipses. Variables are assigned by means of the assignment operator =. The expression

```
(i=3)
```

will assign the value 3 to the variable `i`. Variable names consist of strings no longer than 8 characters. Assignment expressions do not return any result, but an error is reported if the evaluation is incorrect. Arithmetic functions are defined similarly:

```
(f(x)=a*x+b)
```

assigns the expression “a*x+b” to the function “f(x)”. Note that all variables are global. There are no local variables. Arguments to functions must have different denominations when defined and when referenced (due to a bug).

Expressions which contain no equal sign return the result, i.e. the expression evaluator substitutes the expression by its result. Note that variables whose names start by *i-n* or *I-N* return integer expressions if the result of the evaluation is to be transmitted to the application (see above example).

Due to a bug, real variables in the form of

`(a=.111)` or `(a=-.111)`

must be preceded by 0:

`(a=0.111)` or `(a=-0.111)`

The interpreter cannot access and handle vectors or vector elements.

Flow control statements

A limited number of flow control statements are available, such as *loop*, *while*, *break*, *exit* and *if*. Flow control statements are evaluated if the matching ellipse is found. The sequence

```
(i=1) while ( i<=10 ) ( loop_body_commands (i=i+1 ) )
```

is placed on the loop stack. Internally the loop body is augmented by

```
( nbreak ( i<=10 ) loop_body_commands (i=i+1) )
```

The loop described in the above example is executed until i is greater than 10, i.e. until *nbreak* is “true”, or until the maximum loop count has been reached. The maximum loop count provides for additional security, for instance during debugging of a script. It can be set by the

```
set loopcount maxval
```

command. Initially, the maximum loop count is set to “infinity”, i.e. 999999999. The *eval* instruction evaluates the expression placed between ellipses and prints the result.

Procedures

Procedures or scripts are collections of statements placed in a text file which resides in the current directory. Procedures are invoked by name:

```
@proc_name (p1 p2 ...)
```

The procedures parameters pi are optional (1 to 9 parameters). Parameters within the procedures must be referenced by $1 $2 etc.

Since the command language does not allow for local variables all variables within the procedures are global.

Interpreter Commands

Arithmetic Expressions

Arithmetic expressions allow for the assignment (definition) and evaluation of variables and arithmetic functions. Arithmetic expressions must be placed between ellipses. All calculations within the arithmetic module are performed in double precision. The interpreter can only handle scalar values.

Assignment operations return no result of the evaluation of the expression. Any evaluation of an expression returns a single scalar floating-point result, except for variables of function names starting with *i-n* or *I-N*.

Synopsis

(name=value)

(name)

Description

The assignment expression (name=value) assigns the function “value” to the variable “name”. The evaluation expression (name) evaluates the function “name” and returns the result.

Arithmetic expressions and functions are formulated according to *Fortran* conventions. Indices, i.e. references to vector elements, cannot be handled by the interpreter. Some built-in mathematical functions are available:

Trigonometric functions	sin(), cos(), asin(), acos(), tan(), atan().Arguments to trigonometric functions are in radians.
Square root	sqrt()
Logarithmic functions	log(), log10()
Exponentiation	exp()

Note: Real numbers may not start by the dot "." character due to a bug, but they must be preceded by a leading 0:

```
(a=.123)
```

must be replaced by

```
(a=0.123)
```

and the expression

```
(a=-.123)
```

must be replaced by

```
(a=-0.123)
```

Example: Assign the value “23” to the integer variable “i”

```
(i=23)
```

Example: Define a function which transforms polar coordinates (r,phi) to cartesian coordinates (x,y)

```
(drad=3.14159/180.)
(fx(r,phi)=r*cos(drad*phi)
(fy(r,phi)=r*sin(drad*phi)
```

Example: Evaluate functions fx and fy for r=10. and phi=30.

```
eval (fx(10.,30.)) eval (fy(10.,30.))
```

Flow control statements

BREAK NBREAK EXIT - Interrupt loop

break interrupts a loop (*loop* command) if the logical expression *condition* is true. *nbreak* interrupts a loop (*loop* command) if the logical expression *condition* is false. *exit* unconditionally interrupts a *loop* or an *if* body.

Synopsis

break (condition)

nbreak (condition)

exit

Description

The conditional expression *condition* must be placed between ellipses and it may only contain logical operators in the form

```
variable operator variable
```

The > operator means “greater than”. The = operator means “greater, or equal to”. The < operator means “smaller than”. The <= operator means “smaller than, or equal to”. The **|** operator means “or” (exclusive “or”). The **&** operator means “and”.

The expression may contain constants and variables. Variables whose names start by *i-n* or *I-N* are interpreted as integers.

Example

The expression

```
break ( i>10 )
```

stops the loop if the variable `i` is greater than 10.

I F - Conditional execution

if executes a sequence of commands if the logical expression *condition* is true.

Synopsis

if (condition) (statements)

Description

The expression *condition* must be placed in ellipses and it may only contain logical operators in the form

```
variable operator variable
```

The > operator means “greater than”. The = operator means “greater, or equal to”. The < operator means “smaller than”. The <= operator means “smaller than, or equal to”. The **|** operator means “or” (exclusive “or”). The **&** operator means “and”.

The expression may contain constants and variables. Variables with names starting by *i*-*n* or *I*-*N* are interpreted as integers.

LOOP - Unconditional loop

loop executes instructions following the *loop* command until a *break* instruction placed within the loop will be true, or until the maximum loop count (see *set loopcount*) has been reached.

Synopsis

loop (loop_body)

Description

Any commands and expressions may be placed in the loop body. However, *loop* interprets commands without taking into account end-of-line marks. It is therefore necessary to explicitly separate commands by a semicolon (;). The semicolon is not always required, but it is safer to specify it as a rule. Variables with names starting by *i-n* or *I-N* are interpreted as integers.

Example

Execute a loop under the control of the variable *i*. If *i* is greater than *10*, interrupt the loop.

```
(i=1)
loop (
  eval (i) (i=i+1)
  if ( i > 10 ) ( break; )
  )
```

WHILE - Conditional loop

while executes instructions placed in the loop body until *condition* will be true, or until the maximum loop count (see *set loopcount*) is reached. *while* performs the same function as *loop* with the exception that the *break* command is specified separately.

Synopsis

while (condition) (body)

Description

Any commands and expressions may be placed within the body. However, *while* interprets commands without taking into account end-of-line marks. It is therefore necessary to explicitly separate commands by a semicolon (;). The semicolon is not always required, but it is safer to specify it as a rule. Variables with names starting by *i-n* or *I-N* are interpreted as integers.

Example

Execute a loop under the control of the variable *i*. If *i* is greater than *10*, interrupt the loop.

```
(i=1)
while ( i <= 10) (
  eval (i) (i=i+1)
  )
```

Output commands

ECHO - Print string

echo prints a string. *echo* is useful if included in procedures (scripts) or in loops.

Synopsis

echo "string"

Description

The string may be composed of any printable characters delimited by other characters not contained in the string. Echo strings are not evaluated.

EVAL - Evaluate expression

eval evaluates an arithmetic expression and prints the result. The expression must be placed between ellipses.

Synopsis

eval (expression)

Description

The expression must be a valid arithmetic expression. *eval* cannot evaluate application commands.

Data base variables

Variables stored on a data base file may be accessed within the interpreter. Currently, only variables stored in relational tables and data set descriptors can be accessed.

Synopsis

gettab table

getdes table

puttab table

putdes table

extract key

insert key

Description

gettab and *getdes* open the relation table or data set descriptor `table` for access by the *extract* and *insert* commands. *puttab* and *putdes* close the table and write the (modified) contents to the data base. *extract* copies the contents of the variable `key` to the local variable stack under the same name and data type. *insert* copies the local variable `key` to the table.

Example

Load the ***ADIR*** data set descriptor table and extract the number of sub-domains (***NSD***). Print the contents of ***NSD***.

```
getdes ADIR; extract NSD; eval (NSD)
```

Bugs

Arrays cannot yet be manipulated.

Data Base File Manager Commands

OPEN - Open data base file

open opens a single *MemCom* data base file. The data base file is opened and assigned to unit ***iu***. If the operation is successful, this file becomes the primary file. All operations are related to unit ***iu***, unless another file unit becomes the primary unit (*unit* command). The data base file remains open throughout the *Monitor* session unit explicitly closed.

Synopsis

open file ***fname*** *unit* ***iu*** *(new | old | scratch | unknown)*

Parameters

file ***fname***
Specifies the name of the data base file to be opened. The file name may not contain blanks or special characters, i.e. ASCII characters 0 to 31.

mode iomode
Specifies the file i/o mode. ***unbuffered*** opens a file in unbuffered (direct) i/o mode (default). ***buffered*** opens a file in buffered (paged) i/o mode. Note that certain operations, like *art* operations, require buffered i/o mode.

new
Opens a new file.

old
Opens an old (existing) file.

scratch
Creates a new file and deletes it after `close` or after the *Monitor* has been left.

unit ***iu***
*S*pecifies the logical file unit ***iu*** which will be assigned to the file ***fname***. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit will be assigned to the file ***fname***.

unknown
Opens an old file if it exists, otherwise opens a new file (default).

Example

Open a new data base file ***demo***. The system returns the file unit associated to the file ***demo.db***.

```
open unit 1 file demo.db new
```

UNIT - Set primary unit

unit specifies the primary file unit. All data set manipulations will refer to the primary file unit unless the *-u* attribute is specified. If the *unit* command is not issued, all data sets not explicitly referring to a logical unit will be assigned to *unit 0*. This also means that data sets referring to unit 0 are deleted when the Dynamic Common is cleared or when the *Monitor* is left.

Synopsis

unit ***iu***

Parameters

iu

Specifies the logical file unit number to become the primary unit. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command).
Default: *unit 0* (or the last unit specified or assigned during an *open* operation).

CLOSE - Close data base file

Closes the data base file assigned to unit ***iu***. If the unit parameter is omitted, all files will be closed. Data sets in DC referring to the unit(s) to be closed are not saved on the data base file(s). To save sets in DC, type *save*.

Synopsis

close (unit **iu** | ***all****)*

Parameters

If no parameter is specified the current unit is closed.

unit **iu**
The *close* operation is performed on unit ***iu***. ***iu*** must be in the range of 1 to 11.

all
Close all active units.

DIR - Print directory of data base file

directory or *dir* displays the data set names and data set attributes of the contents of a data base file. Note that the first 20 characters of the data set names are printed. To get data set names longer than 20 characters printed, specify *full* (see below).

Synopsis

directory ***setname*** *(-u* ***iu****) (short | full | dc)*

Parameters

setname
Specifies the data set name of the data set to be listed. The data set name must be delimited by blanks or by double quotes, and it may not contain blank characters. “Wild-cards” in the data set name specification are permitted according to the rules described earlier in this Chapter.

short
Lis*ts a reduced table of contents.*

full
Lists the full table of contents of the data base file (default).

dc
Lists the full table of contents of the Dynamic Common.

Data set name attributes

-u ***iu***
-u specifies the logical file unit ***iu*** of the data set. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit will be selected (as defined by the *unit* command, or by the first *open* command issued during a session).

Example

Print directory of all sets ***DISP*** on data base file of unit 3:

```
dir DISP* -u 3
```

Data Operation Commands

Data operation commands apply to data sets. Data sets and their attributes are specified by the sequence

name -attribute1 ... -attribute2 ... etc.

Data set names and attribute keywords are case-sensitive. The Monitor data set attributes are listed below:

-d
Designates a data set as discontinuous. -d applies to the *dss*, *input* and *print* commands.

-l ***n1(:n2(:n3))***
-l specifies the number of logical elements of the set to be processed. If *-l* ***n1*** is specified, elements from ***1*** to ***n1*** will be selected. If *-l* ***n1:n2*** is specified, elements from ***n1*** to ***n2*** will be selected. If *-l* ***n1:n2:n3*** is specified, elements from ***n1*** to ***n2*** in steps of ***n3*** will be selected. The *-e* attribute is equivalent to *-l*.
Default: *-l* ***1***.

-s ***idiv***
-s specifies the sub-division number ***idiv*** of the data set (if any). Sub-division numbers are only valid for data sets with sub-divisions.
Default value for ***idiv***: 0 (no sub-division specified).

-t ***dtype***
-t specifies the data set type ***dtype*** (see Appendix for a description of the data set types). Data set types are not case-sensitive, i.e. they are automatically transformed to upper-case.
Default: System-dependent or set by the *set type* command.

-v ***value***
-v specifies an initialization value for each element of a new set. The value must correspond to the data type ***dtype***.

-u ***iu***
-u specifies the logical file unit ***iu*** of the data set. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit will be selected (as defined by the *unit* command, or by the most recent *open* command).

Some Monitor commands, such as *directory* or *print*, allow for operations on incomplete data set names, also referred to as “wild-card” operations. For example, to print all sets `DISP.ext`, where `ext` is any string attached to `DISP`, type

```
print DISP.*
```

Observe the following wild-card rules:

set name	Description
*	All data sets are included in the command (default).
text	The data set `text` is included.
text*	All data sets whose labels start by `text` are included.
text	All data sets containing the string `text` somewhere in the data set name are included.

INPUT - Enter data

The *input* command inserts data interactively into a set. Data is accepted from the current input device. The data types *I, E, F, D* and *A* are supported. To insert data in relational tables, make use of the table subsystem command.

To load formatted or binary data from a file, use *read*.

Synopsis

input ***setname*** *(attributes)*
[...data...]

Parameters

setname
Specifies the data set name. The data set name must be delimited by blanks or by double quotes, and it may not contain blank characters.

Data set name attributes

-s ***idiv***
-s specifies the sub-division number of the data set. ***idiv*** must be in the range of 0 to ***ndiv***, where ***ndiv*** is the number of sub-sets of the current data set ***setname***. Specification of the sub-division index ***idiv*** is valid only if the data set has been declared as a super-set (see ***dss*** command). Default value for ***idiv*** is 0, i.e. no sub-division.

-t ***dtype***
Specifies the data type of the object ***key***. The object type ***dtype*** may consist of `I` (integer), `E` (real), `D` (double precision), `F` (real*8) or `K` (character). The object type is not case-sensitive. The default data type is installation dependent and it may be specified by the ***set*** command.

-l ***n1(:n2(:n3))***
-l specifies the number of logical elements of the set to be processed. If *-l* ***n1*** is specified, elements from ***1*** to ***n1*** will be selected. If *-l* ***n1:n2*** is specified, elements from ***n1*** to ***n2*** will be selected. If *-l* ***n1:n2:n3*** is specified, elements from ***n1*** to ***n2*** in steps of ***n3*** will be selected. The *-e* attribute is equivalent to *-l*.
Default: *-l* ***1***.

-u ***iu***
-u specifies the logical file unit ***iu*** of the data set ***setname***. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit will be selected (as defined by the *unit* command, or by the latest *open* command).

Bugs

1. Due to a problem with the interpreter, one or more negative numbers must always be enclosed in brackets.
2. The *-l* attribute does not work properly for A sets. You must always enter elements from *1* to *n*, i.e. with the parameter *-l n*.

3. Data of type A may not be placed in brackets and they must be listed on a new line. Blanks will be inserted. Example:

```
input enam -l 3 -u 1 -t A
AAAABBBCCCC
```

4. The data type attribute -t is required even in case the set already exists.

Examples

Example 1: Fill data set `colors` on unit 3 with 13 words of data type E.

```
input colors -l 13 -u 3 -t E

[ 1.0 2 3 4 5. 78. 3. 3. 9.E-2 10. ]
```

Example 2: Modify the `N`-th subdivision of data set `COOR.1`. The set has sub-divisions of 3 elements. Note that ellipses evaluate an expression and feed the result into the input stream.

```
# define N (example)
(N=24)

input COOR.1 -s (N)

[ 12.5 9.0 -34.9 ]
```

Thus, the 3 columns of the 24th row are modified.

READ - Read from external file

read copies data from an external text file to a data set on a data base file. If the data is to be typed from the terminal, use the `input` command. `read` is only valid for positional data, i.e. data of type *A, C, D, E, F, I* and *K*. Relational tables are manipulated by the *table* system.

Synopsis

read ***setname*** *(attributes) (file* ***fname****)*

Parameters

setname
Specifies the data set name of the data set to be read. The data set name must be delimited by blanks or by double quotes, and it may not contain blank characters.

file ***fname***
File name of the data input file. The default file ***monitorin.txt*** will be read if *file* is omitted.

Data set name attributes

-l ***nel***
-l ***nel*** specifies the number of logical elements of the set to be read.
Default: *-l* ***1***.

-s ***idiv***
-s specifies the sub-division number. ***idiv*** must be in the range of 0 to ***ndiv***, where ***ndiv*** is the number of sub-sets of the current data set ***setname***. Specification of the sub-division index ***idiv*** is valid only if the data set has been declared as a super-set (see ***dss*** command). Default value for ***idiv*** is 0, i.e. no sub-division.

-t ***dtype***
-t specifies the data set type ***dtype*** (see Appendix for a description of data set types). Data set types are not case-sensitive, i.e. they are transformed to upper-case. The default value is system-dependent or is set by the *set type* command.

-u ***iu***
-u specifies the logical file unit ***iu*** of the data set to be read. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit will be selected (as defined by the *unit* command, or by the latest *open* command).

Example

Read 2000 words in double precision format from formatted file ***datafil.txt*** and store it in data set ***XYZ***:

```
read XYZ -l 2000 -t D datafil.txt
```

WRITE - Write to external file

write copies the contents of a data set to an external text file. *write* is only valid for positional data, i.e. data of type *A, C, D, E, F, I* and *K*.

Synopsis

*write **setname** (attributes) (file **fname**)*

Parameters

setname
Specifies the data set name of the data set to be written. The data set name must be delimited by blanks or by double quotes, and it may not contain blank characters.

*file **fname***
Specifies the ASCII (text) file ***fname*** to which values are written. The default file ***monitorout.txt*** will be created if *file* is omitted.

Data set name attributes

*-u **iu***
-u specifies the logical file unit ***iu*** of the data set. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit is selected (as defined by the *unit* command or the latest *open* command).

*-s **idiv***
Specifies the sub-division index. ***idiv*** must be in the range of 0 to ***ndiv***, where ***ndiv*** is the number of sub-sets of the current data set ***setname***. Specification of the sub-division index ***idiv*** is valid only if the data set has been declared as a super-set (see ***dss*** command). Default value for ***idiv*** is 0, i.e. no sub-division.

*-t **dtype***
Specifies the data type of the object ***setname***. The object type ***dtype*** may consist of `I` (integer), `E` (real), `D` (double precision), `F` (real*8) or `K` (character). The object type is not case-sensitive. The default data type is installation dependent, and it may be specified by the ***set*** command.

*-l **nelements***
Number of elements in the object ***key***. If l is omitted, a single data element is assumed.

Example

Write 2000 words from data set ***XYZ***, subdivision 4, in double precision format to file ***newfil.dat***:

```
write XYZ -s 4 -t d -l 2000 newfil.dat
```

PRINT - Print content of data set

The *print* command prints the content of a data set. A data set is printed according to the print format associated to the data set (see *set format* command). The *-t* attribute overrides the data set type and prints the set according to the format associated to the type specified by *-t*. Data are printed on the current output device (default: terminal). Output may be redirected by *print* parameters or by the *set output* command (see *set*). *print* is only valid for positional data, i.e. data of type A, C, D, E, I.

Synopsis

print ***setname*** *(attributes)*

Parameters

setname

Specifies the data set name of the data set to be printed. The data set name must be delimited by blanks or by double quotes, and it may not contain blank characters. “Wild-cards” in the data set name specification are permitted according to the rules described earlier in this Chapter.

Data set name attributes

-l ***n1(:n2(:n3))***

-l specifies the number of logical elements of the set which are to be processed. If *-l* ***n1*** is specified, elements from ***1*** to ***n1*** are selected. If *-l* ***n1:n2*** is specified, elements from ***n1*** to ***n2*** are selected. If *-l* ***n1:n2:n3*** is specified, elements from ***n1*** to ***n2*** in steps of ***n3*** are selected. The *-e* attribute is equivalent to *-l*.
The default print size of the set is the set size or 100 logical elements, whichever is smaller.

-s ***idiv***

-s specifies the sub-division number ***idiv*** of the data set(s) to be printed. ***idiv*** must be in the range of 0 to ***ndiv***, where ***ndiv*** is the number of sub-sets of the current data set ***setname***. Specification of the sub-division index ***idiv*** is valid only if the data set has been declared as a super-set (see ***dss*** command). Default value for ***idiv*** is 0, i.e. no sub-division is selected.

-t ***dtype***

Specifies the print data type of the object ***setname***. The object type ***dtype*** may consist of `I` (integer), `E` (real), `D` (double precision), `F` (real*8) or `K` (character). The object type is not case-sensitive. The default data type is the one of the data set to be printed.

-u ***iu***

-u specifies the logical file unit ***iu*** of the data set. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit is selected (as defined by the *unit* command or the latest *open* command).

Examples

Print data set ***TEMP.5*** stored on unit 3. The number of words printed is defined by the data set (the -l or -e attribute has been omitted).

```
print TEMP.5 -u 3
```

Print all data set ***COOR*** with name suffixes ***.1***, ***.2***, etc..

```
print COOR.*
```

MINMAX - Print minimum and maximum values of data set

The *minmax* command prints the minimum and maximum values of a data set. *minmax* command is only valid for positional data of type *D*, *E*, *I*, and *F*.

Synopsis

minmax ***setname*** *(attributes)*

Parameters

setname
Specifies the data set name of the data set to be scanned. The data set name must be delimited by blanks or by double quotes, and it may not contain blank characters. “Wild-cards” in the data set name specification are permitted according to the rules described earlier in this Chapter.

Data set name attributes

-l ***n1(:n2(:n3))***
-l specifies the number of logical elements of the set which have to be processed. If *-l* ***n1*** is specified, elements from ***1*** to ***n1*** are selected. If *-l* ***n1:n2*** is specified, elements from ***n1*** to ***n2*** are selected. If *-l* ***n1:n2:n3*** is specified, elements from ***n1*** to ***n2*** in steps of ***n3*** are selected. The *-e* attribute is equivalent to *-l*.
The default print size of the set is the set size or 100 logical elements, whichever is smaller.

-s ***idiv***
-s specifies the sub-division number ***idiv*** of the data set(s) to be printed. ***idiv*** must be in the range of 0 to ***ndiv***, where ***ndiv*** is the number of sub-sets of the current data set ***setname***. Specification of the sub-division index ***idiv*** is valid only if the data set has been declared as a super-set (see ***dss*** command). Default value for ***idiv*** is 0, i.e. no sub-division.

-u ***iu***
-u specifies the logical file unit ***iu*** of the data set. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit is selected (as defined by the *unit* command or the latest *open* command).

RESE - Create data set on data base

rese creates a data set on a data base file. The set is initialized with a constant value for each element of the set. Valid data types are *I, E, F, D, A,* and *$*.

Synopsis

rese ***setname*** *(attributes)*

Parameters

setname
Specifies the data set name of the data set to be created. The data set name must be delimited by blanks or by double quotes, and it may not contain blank characters.

const ***value***
Specifies an initialization value for each element of a new set. The value must correspond to the data type ***dtype***.

Data set name attributes

-l ***n1***
-l specifies the size of the set which is to be created.
Default: *-l* ***1***.

-t ***dtype***
-t specifies the data set type ***dtype*** of the set to be created. Data set types are not case-sensitive. They are transformed to upper-case.
Default: System-dependent or set by *set type*.

-u ***iu***
-u specifies the logical file unit ***iu*** of the data set. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit is selected (as defined by the *unit* command or the latest *open* command).

DSS - Create sub-divided data set on data base

dss creates a data set with sub-divisions. The set is initialized with a constant value for each element of the set. Valid data types are *I, E, F, D, A,* and *$*.

Synopsis

dss ***setname*** *(attributes)*

Parameters

setname
Data set name of set to be created. The data set name may not contain blank characters.

const ***value***
Specifies an initialization value for each element of a new set. The value must correspond to the data type ***dtype***.

Data set name attributes

-d
Flags a discontinuous data set.

-l ***n1***
-l specifies the size of a sub-set which is to be created.
Default: *-l* ***1***.

-s ***ndiv***
-s specifies the number of sub-divisions ***ndiv*** of the data set to be created.

-t ***dtype***
-t specifies the data set type ***dtype*** of the set to be created. Data set types are not case-sensitive. They are transformed to upper-case.
Default: System-dependent or set by *set type*.

-u ***iu***
-u specifies the logical file unit ***iu*** of the data set. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit is selected (as defined by the *unit* command or the latest *open* command).

COPY - Copy data set

copy copies a single data set. If the target data set does not exist, *copy* will create it. The target data set ***settarget*** can be renamed.

scopy copies selected data sets to another data base file. The target data set(s) cannot be renamed.

Synopsis

copy ***setsource*** *(attributes)* ***settarget*** *(attributes)*

scopy ***setsource*** *(attributes)* ***** *(attributes)*

Parameters

setsource
Data set name of set to be copied. The data set name may not contain blank characters.

settarget
Data set name of target set. The data set name may not contain blank characters.

Data set name attributes

-u ***iu***
-u specifies the logical file unit ***iu*** of the source or target data set. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit is selected (as defined by the *unit* command or the latest *open* command).

Bugs

- Subsets cannot be copied.
- Wild-cards are not permitted in set name specifications of *copy*. Make use of *scopy* instead.

Example

Copy data set ***DISP.1.1*** on unit 1 to data set ***DISP.1.1*** on unit 11.

```
copy DISP.1.1 -u 1 DISP.1.1 -u 11
```

Copy all data sets ***DISP.**** on unit 1 to unit 11.

```
scopy DISP.* -u 1 * -u 11
```

FCOPY - Copy all data sets

fcopy copies all data sets of the data base file identified by unit ***iu1*** to another data base file identified by unit ***iu2***. Inactive or deleted data sets are not copied. *fcopy* may thus be used to compress (pack) a data base file. Both files ***iu1*** and ***iu2*** must be open (see *open* command).

Synopsis

fcopy ***iu1 iu2***

Parameters

iu1
Logical unit of source data base file. ***iu1*** must be in the range of *1* to *11*.

iu2
Logical unit of target data base file. ***iu2*** must be in the range of *1* to *11*.

Example

Pack a data base file by copying all sets. The operation is performed in three steps: (1) Open data base files, (2) Copy all sets, (3) Rename data base files.

```
open unit 1 file demo.db old
open unit 8 file demo.db.bak new
fcopy 1 8
system "mv demo.db demo.db.old; mv demo.db.bak demo.db"
```

COMPRESS - Compress data base file

compress compresses the data base file identified by unit ***iu***. The compression procedure removes all deleted or inactive data on the data base. Unlike *fcopy*, *compress* operates on the same data base file. It thus does not require more disk space.

Synopsis

compress ***iu***

Parameters

iu

iu specifies the logical file unit ***iu*** of the data base to be compressed. ***iu*** must be in the range of *1* to *11*.

RWA - Read in word-addressable mode

rwa extracts data from a data base file in word-addressable mode and prints them using the format corresponding to the data type specified (see *set format* command). *rwa* provides a low-level access to the data stored on a data base file. Note that the parameters must be specified in the order listed below.

Synopsis

rwa ***iu dtype iadr nwords***

Parameters

iu
Specifies the logical file unit iu. ***iu*** must be in the range of *1* to *11*.

dtype
Data type to be printed. Must be one of A | D | E | F | Z | I (see Appendix for the explanation of data set types).

iadr
Start (word) address of word stream on data base file ***iu***.

nwords
Number of machine words to be extracted and printed.

WWA - Write in word-addressable mode

wwa copies data entered at the terminal to the Data Base file in word-addressable mode. Data is converted according to the data type specified. Note that the parameters must be specified in the order listed below.

Synopsis

wwa ***iu dtype iadr nwords***
...data...

Parameters

iu
Specifies the logical file unit iu. ***iu*** must be in the range of *1* to *11*.

dtype
Data format to be read. Must be one of A | D | E | F | Z | I (see Appendix for the explanation of data set types).

iadr
Start (word) address of word stream on data base file ***iu***.

nwords
Number of machine words to be inserted.

DELETE - Delete data set

delete deletes a data set from data base file. Data sets are not physically removed during delete operations. They are marked for deletion but still figure in the tables. The *fcopy* or the *compress* command physically remove all deleted data sets.

Synopsis

delete ***setname*** *(attributes)*

Parameters

setname
Data set name of set to be deleted. The data set name may not contain blank characters. “Wildcards” in the set specification are permitted according to the rules described earlier in this Chapter.

Data set name attributes

-u ***iu***
-u specifies the logical file unit ***iu*** of the data set to be deleted. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit is selected (as defined by the *unit* command or the latest *open* command).

Example

Delete all data sets ***DISP****, i.e. data sets starting by ***DISP*** and containing any trailing string:

```
delete DISP*
```

RENAME - Rename data set

rename changes the name of a data set on a data base file. Note that the unit attribute of the new data set name is ignored.

Synopsis

rename ***setold*** *(attributes)* ***setnew*** *(attributes)*

Parameters

setold
Data set name of the set to renamed.

setnew
New data set name.

Data set name attributes

-u ***iu***
-u specifies the logical file unit ***iu*** of the data set ***setold*** to be renamed. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit is selected (as defined by the *unit* command or the latest *open* command).

Example

Rename data set ***OLDSET*** to ***NEWSET***

```
rename OLDSET NEWSET
```

The Table Subsystem

table invokes the relational table subsystem. It allows for manipulating relational table data (i.e. data sets of type *$* or data set descriptors). You may

- Load relational tables and descriptors from data base files
- Store relational tables and descriptors to data base files
- Edit relational tables and descriptors

Relational table data sets contain information as described in the *MemCom* Reference Manual. The sets are valid *MemCom* data sets and differ from other data sets only by the data set type *$*. In general, table data sets may only be manipulated through the table buffer using TB routines. The sets are then mapped from the data base files to the table buffer and vice versa (see *MemCom* User Manual for details).

Synopsis

table commands end

Parameters

See all paragraphs of this section. To leave the *table* subsystem, type *end*.

Summary of table manager commands

Name	Description
cflag	Sets-clears the `noclear` flag
clear	Clears table buffer
extract	Extracts data from table
getdes	Gets descriptor record from data base file
gettab	Gets table data set from data base file
insert	Inserts data into table
length	Gets current length of table
putdes	Puts descriptor record back to data base
puttab	Puts table data set back to data base file
print	Prints contents of table
replace	Replace data in table
reserve	Create empty relational table on data base

GETDES - Load data set descriptor table

getdes copies the descriptor of the data set `label` into the table buffer. *getdes* must be invoked if descriptor entries are to be accessed. This command is only operative when in DB mode (see *set mode* command).

Synopsis

getdes ***setname*** *(attributes)*

Parameters

setname

Data set name of the descriptor to be loaded. The data set name starts with the first non-blank character and it ends with a blank or a slash (if attributes are present). “Wildcards” are not permitted.

Data set name attributes

-u ***iu***

-u specifies the logical file unit ***iu*** of the data set ***setname*** whose descriptor is to be loaded. ***iu*** must be in the range of *1* to *11*.

GETTAB - Load relational table

gettab copies the table data set of type *$* from the data base file into the table buffer. *gettab* must be invoked if relational table entries are to be accessed. This command is only operative when in DB mode (see *set mode* command).

Synopsis

*gettab **setname** (attributes)*

Parameters

setname
Data set name of the relational table to be loaded. The data set name starts with the first non-blank character and it ends with a blank or a slash (if attributes are present). “Wildcards” are not permitted.

Data set name attributes

*-u **iu***
-u specifies the logical file unit ***iu*** of the relational table data set ***setname*** to be loaded. ***iu*** must be in the range of *1* to *11*.

*-s **idiv***
-s specifies the sub-division number ***idiv*** of the relational table data set to be loaded (if any). ***idiv*** must be in the range of 0 to ***ndiv***, where ***ndiv*** is the number of sub-sets of the current data set ***setname***. Specification of the sub-division index ***idiv*** is valid only if the data set has been declared as a super-set (see ***dss*** command). Default value for ***idiv*** is 0, i.e. no sub-division.

PUTDES - Write data set descriptor table

putdes transfers the table presently held in the table buffer to the descriptor of the data set ***setname*** on the data base file. The old descriptor on the data base file is destroyed. By default the table buffer will be automatically cleared. This command is only operative when in DB mode (see *set mode* command).

Synopsis

putdes ***setname*** *(attributes)*

Parameters

setname

Data set name of the descriptor to be saved. The data set name starts with the first non-blank character and it ends with a blank or a slash (if attributes are present). “Wildcards” are not permitted.

Data set name attributes

-u ***iu***

-u specifies the logical file unit ***iu*** of the data set ***setname*** whose descriptor is to be saved. ***iu*** must be in the range of *1* to *11*.

PUTTAB - Write relational table

puttab writes the relational table presently in the table buffer to the data base file and assigns it to the data set ***label***. By default the table buffer will be automatically cleared after the *puttab* operation. This command is only operative when in DB mode (see *set mode* command).

Synopsis

puttab ***setname*** *(attributes)*

Parameters

setname

Data set name of the relational table to be saved. The data set name starts with the first non-blank character and it ends with a blank or a slash (if attributes are present). “Wildcards” are not permitted.

Data set name attributes

-u ***iu***

-u specifies the logical file unit ***iu*** of the data set ***setname*** which is to be saved. ***iu*** must be in the range of *1* to *11*.

-s ***idiv***

-s specifies the sub-division number ***idiv*** of the relational table data set to be written to the data base file (if any). ***idiv*** must be in the range of 0 to ***ndiv***, where ***ndiv*** is the number of sub-sets of the current data set ***setname***. Specification of the sub-division index ***idiv*** is valid only if the data set has been declared as a super-set (see ***dss*** command). Default value for ***idiv*** is 0, i.e. no sub-division.

RESERVE - Create empty relational table on data base

reserve creates a new relational table on a data base file. The table is empty. This command is only operative when in DB mode (see *set mode* command).

Synopsis

reserve ***setname*** *(attributes)*

Parameters

setname

Data set name of the relational table data set to be created. The data set name starts with the first non-blank character and it ends with a blank or a slash (if attributes are present). “Wildcards” are not permitted.

Data set name attributes

-l ***size***

-l specifies the size in words of the relational table or of a sub-set of the relational table. If ***size*** is omitted, the maximum table size will be selected.

-n ***ndiv***

-n specifies the number of sub-divisions ***ndiv*** of the data set (if any). Specification of the number sub-divisions ***ndiv*** creates a super-set. Default value for ***ndiv*** is 0.

-u ***iu***

-u specifies the logical file unit ***iu*** of the table ***setname*** which is to be created. ***iu*** must be in the range of *1* to *11*.

INSERT - Insert data in relation table

insert inserts the object ***key*** into the table buffer. *insert* modifies the current table in the table buffer. To save the modified table on the data base, make use of the *puttab* command (for relational tables) or the *putdes* command (for data set descriptors). This command is only operative when in DB mode (see *set mode* command).

Synopsis

*insert **key** (attributes) **[values]***

Parameters

key
Name of the object to be inserted into the table buffer.

value
Value of the object to be inserted into the table buffer. The value must correspond to the object data type (see below). If the object consists of an array of values, place the values between brackets (***I*** (integer), ***E*** (real), ***D*** (double precision), ***F*** (real*8) data types only). ***K*** (character) data strings must be enclosed by quotes or double quotes if the string contains blank characters.

Key attributes

*-t **dtype***
Specifies the data type of the object ***key***. The object type ***dtype*** may consist of ***I*** (integer), ***E*** (real), ***D*** (double precision), ***F*** (real*8) or ***K*** (character). The object type is not case-sensitive.

*-l **size***
Number of elements in the object ***key***. If ***size*** is omitted, a single data element is assumed.

Bugs

Due to a problem with the interpreter, one or more negative numbers must always be enclosed within brackets. Example:

```
insert MINVAL -t F -l 3 [ -1.0e30 -1.0e30 -1.0e30 ]
```

EXTRACT - Extract data from relational table

extract extracts the object ***key*** from the table or from the descriptor present in the table buffer and prints it. This command is only operative when in DB mode (see *set mode* command).

Synopsis

*extract **key** (attributes)*

Parameters

key
Name of the entry to be extracted from the table buffer. The entry is printed at the terminal.

Key attributes

*-l **size***
Number of elements in the object ***key***. If ***size*** is omitted, a single data element is extracted.

REPLACE - Replace data in relational table

replace modifies the object ***key*** residing in the table buffer. If an object does not exist in the table it will be created by *replace*. *replace* modifies the current table in the table buffer. To save the modified table on the data base, make use of the *puttab* command (for relational tables) or the *putdes* command (for data set descriptors). This command is only operative when in DB mode (see *set mode* command).

Synopsis

replace ***key*** *(attributes)* ***value***

Parameters

key

Name of the object to be replaced in the table buffer.

value

Value of the object to be replaced. The value must correspond to the object data type (see below). If the object consists of an array of values, place the values between brackets (***I*** (integer), ***E*** (real), ***D*** (double precision), ***F*** (real*8) data types only). ***K*** (character) data strings must be enclosed by quotes or double quotes if the string contains blank characters.

Key attributes

-t ***dtype***

Specifies the data type of the object ***key***. The object type ***dtype*** may consist of ***I*** (integer), ***E*** (real), ***D*** (double precision), ***F*** (real*8) or ***K*** (character). The object type is not case-sensitive.

-l ***size***

Number of elements in the object ***key***. If ***size*** is omitted, a single data element is assumed.

Bugs

Due to a problem with the interpreter, one or more negative numbers must always be enclosed in brackets. Example:

```
replace MINVAL -t F -l 3 [ -1.0e30 -1.0e30 -1.0e30 ]
```

DELETE - Delete data from relational table

delete deletes an object from the table buffer. To save the modified table, make use of the *puttab* command (for relational tables) or the *putdes* command (for data set descriptors). This command is only operative when in DB mode (see *set mode* command).

Synopsis

delete ***key***

Parameters

key
Name of the object to be deleted from the table buffer.

PRINT - Print data of relational table

print prints all or selected objects of the current table. This command is only operative when in DB mode (see *set mode* command).

Synopsis

print ***(key)*** *(attributes)*

Parameters

key
Name of the object to be printed. If ***key*** is omitted, all objects residing in the table will be printed. “Wildcards” in the name specification are permitted according to the rules described earlier in this Chapter.

Key attributes

-l ***nelements***
Number of elements of the object ***key*** to be printed. If omitted, all elements of the object ***key*** will be printed.

CLEAR - Clear relational table

Clears the table buffer. All objects residing in the table buffer are destroyed.

Synopsis

clear

CFLAG - Set clear flag

cflag sets the no-clear flag which controls the resetting of the table buffer after a table has been written to a file. If *noclear* flag is ***off***, the buffer will be cleared, if *noclear* flag is ***on***, the buffer will not be reset after *putdes* or *puttab*. This command is only operative when in DB mode (see *set mode* command).

Synopsis

cflag ***on-off***

LENGTH - Size of relational table

length prints the current size of the table. The size is counted in number of machine words. This command is only operative when in DB mode (see *set mode* command).

Synopsis

length

The Neutral File Manager

The *Monitor* Neutral File Manager (*net*) provides basic tools for formatting data residing on *MemCom* data base files. Formatted files can easily be interchanged between platforms of different binary format. *net* performs the same operations as the corresponding *MemCom* procedures (See *MemCom Reference Manual*, Chapter *Network Manager (NM)*.

net formats data bases into machine-independent format (ASCII format). *net* does not support the actual transport of data between different platforms. Data exchange has to be carried out by external utilities, such as TCP-IP. The basic steps of data exchange consist of:

1. Extraction of data from a data base saving them in machine-independent format on a file.
2. Transport of the file by external utilities, such as TCP-IP.
3. Insertion of the information on the file into the target data base on the target machine.

Steps 1 and 3 are automatically executed by *net*. Step 2 must be controlled by the user.

Synopsis

net parameters end

Parameters

See all paragraphs of this section. To leave the *net* subsystem, type *end*.

Summary of *net* commands

Command	Description
initnm	Initialize *net* system
opnf	Open neutral formatted file
clnf	Close neutral formatted file
send	Send data set(s) to neutral formatted file
retr	Receive data set(s) from neutral formatted file
end	Leave *net* subsystem
deflag	Double precision to single precision conversion flag
edflag	Single precision to double precision conversion flag
dire	Print directory of data base

The Neutral File Manager does not recognize all *MemCom* data set types. The following data set types are currently supported by the Neutral File Manager:

Table 1 *net* data types

A	Alphanumeric
C	Complex
D	Double precision
E	Real
F	Float (Fortran: real*8, C: double)
I	Integer

Table 1 *net* data types

A	Alphanumeric
K	CHARACTER-type text
P	Procedures, ASCII text files
$	Relational tables

The Neutral File Manager subsystem is invoked by typing *net*. Type *end* to leave the subsystem. Data base *open* and *close* functions must be performed in the Monitor command module.

OPNF - Open neutral file

opnf opens a neutral file. All subsequent *send* and *retrieve* operations are performed to and from the neutral file.

Synopsis

opnf ***fname (posflag)***

Parameters

fname
: Name of neutral file. If a file already exists, it will be overwritten.

posflag
: File pointer position flag. The file ***fname*** is rewound if ***flag*** is set to ***R***. The file ***fname*** is not rewound if ***flag*** is set to ***N***, i.e. data are appended to the file (default).

CLNF - Close neutral file

clnf closes the neutral file. The neutral file is automatically closed when the *net* subsystem is left, i.e when the *end* command is issued.

Synopsis

clnf

SEND - Write to neutral file

send copies data sets from the current primary data base to the current neutral file only. Whole data sets may be copied only.

Synopsis

send ***set***

Parameters

set
Data set(s) to be written to a neutral file. “Wild-cards” in the set specification are permitted according to the rules described earlier in this Chapter.

Example

Open neutral file and send all data sets ***DISP.**** to the neutral file ***sfile.txt***.

```
net
  opnf sfile.txt
  send DISP.*
  end
```

RETRIEVE - Read from neutral file

retrieve reads one or more data sets from the current neutral file and inserts them in the current primary data base. A data base file and the neutral file must be open in order to perform the operation.

Synopsis

retrieve ***setname***

Parameters

setname

Data set name of set to be retrieved from the neutral file. Wild-cards in the set specification are permitted according to the rules described earlier in this Chapter. If ***setname*** is set to *******, all sets are read from the neutral file.

Examples

Example 1: Open neutral file and retrieve all sets from neutral file `sfile.txt`.

```
net
  opnf sfile.txt
  retr *
  end
```

Example 2: Open neutral file and retrieve all sets `DISP.*` from neutral file `sfile.txt`.

```
net
  opnf sfile.txt
  retr DISP.*
  end
```

REPFLAG - Data set replace flag

repflag sets the target data base replace flag (*retrieve* only). By default, data sets are not replaced, i.e if a data set already exists, an error message is issued.

Synopsis

repflag ***flag***

Parameters

flag

Replace flag. If flag is set to ***yes***, data sets received from the neutral file will be replaced (default). An error message is issued, if the existing set is shorter than the replacement set. If flag is set to ***no***, data sets will not be replaced. A warning message appears, if the set already exists. If flag is set to ***ovr***, existing data sets will be deleted and replaced by the new set (override flag).

DEFLAG - Convert double to single precision

deflag sets the double precision to single precision conversion flag. The conversion is performed when writing to the neutral file or when reading from the neutral file.

Synopsis

deflag ***flag***

Parameters

flag

Double precision to single precision conversion flag. If ***flag*** is set to ***D-D***, double precision sets (D) will not be converted to single precision (E). If ***flag*** is set to ***D-E***, double precision (D) sets will be converted to single precision (E).

EDFLAG - Convert single-double precision

edflag sets the single precision to double precision conversion flag. The conversion is performed when writing to the neutral file or when reading from the neutral file.

Synopsis

edflag ***flag***

Parameters

flag

Single precision to double precision conversion flag. If ***flag*** is set to ***D-D***, single precision sets (D) will not be converted to double precision (E). If ***flag*** is set to ***D-E***, single precision (D) sets will be converted to double precision (E).

Example

Retrieve data sets ***DISP.**** from a 64 bit word machine to a 32 bit machine. Data is converted from the 64 bit single precision (E) format to the 32 bit single precision (E) format.

```
net
  opnf sfile.txt
  edflag E-E
  retr DISP.*
  end
```

The Array table subsystem

The *art* array table subsystem manipulates *MemCom* array tables.

Synopsis

art parameters end

Parameters

See all paragraphs of this section. To leave the *art* subsystem, type *end*.

Summary of the art commands

Command	Description
open	Open an existing array table.
close	Close an open array table.
init	Create an array table.
print	Print the table of contents of an array table.
extract	Extract data from an array table.
insert	Insert data in an array table.
delete	Delete data in an array table.

OPEN - Open array table

open opens an existing array table on channel ***ichan*** for access by the array subsystem.

Synopsis

*open **tname** (-u **iu)** ch **ichan***

Parameters

tname
Specifies the name of the array table to be opened.

*ch **ichan***
*S*pecifies the logical channel ***ichan*** which will be assigned to the table ***tname***. *ch* is required.

Description of the attributes

*-u **iu***
-u specifies the logical file unit ***iu*** of the table ***tname***.

CLOSE - Close array table

close closes an open array table and frees the assigned channel.

Synopsis

*close (ch **ichan**)*

Parameters

*ch **ichan***
*S*pecifies the logical channel ***ichan*** to be closed. If ***ichan*** is set to 0, all channels are closed. The current channel is closed if *ch* is omitted.

INIT - Create a new table

init creates a new array table on data base specified by ***iu***. The table is not yet open for access (see *open* command).

Synopsis

*init **tname** (-u **iu**) nlines **nl** ncols **nc** colnam **'nam1.name2. ...namen'** space **nwords***

Parameters

tname
Specifies the name of the array table to be created.

Description of the attributes

colnam **'s1.s2.s3...'**
Defines the column names in the form of strings separated by dots.

nlines **nl**
Defines the number of lines.

ncol **nc**
Defines the number of columns.

space **nwords**
Defines the size of the table in words.

PRINT - Print table of contents

print prints a table of contents of the table attached to channel ***ichan***.

Synopsis

*print (ch **ichan**) (l **l1(:l2(:l3)))***

Description of the attributes

*ch **ichan***
Specifies the logical channel ***ichan*** of the table to be printed. If *ch* is omitted, the current table is printed.

*lines **l1(:l2(:l3))***
Prints lines from ***l1*** to ***l2*** in steps ***l3***. If *lines* is omitted, all lines are printed.

ATTRIBUTES - Print attributes of tables

attributes prints the attributes of all open tables.

Synopsis

attributes

EXTRACT - Extract content of a table cell

extract extracts and prints the content of the cell *(iline,icol)* or *(iline,colnam)* of the table attached to ***ichan***. The line is specified by number. The column is specified by number (*cnum* attribute) or by name (*cname* attribute).

Synopsis

*extract (ch **ichan**) (l **line**) (cnum **icol** | cname **cn**) (type **dtype**) (n **nel**)*

Description

*ch **ichan***
: Specifies the channel number ***ichan***.

*line **lnumber***
: Specifies the line number ***lnumber***.

*cnum **ic** | cname **cn***
: Specifies the column. *cnum* specifies the column by number ***ic***. *cname* specifies the column by name ***cn***.

*type **dtype***
: Specifies the data type. Legal data types are I, E, D, and K.

*n **nel***
: Specifies the number data elements to be extracted and printed.

INSERT - Insert data in a table cell

insert inserts data in a cell of the table attached to channel ***ichan***. The line is specified by number. The column is specified by number (*cnum* attribute) or by name (*cname* attribute).

Synopsis

*insert (ch **ichan**) (l **line**) (cnum **icol** | cname **cn**) (type **dtype**) elem **list***

Description

*ch **ichan***
Specifies the channel number ***ichan***.

*line **lnumber***
Specifies the line number ***lnumber***.

*cnum **ic** | cname **cn***
Specifies the column. *cnum* specifies the column by number ***ic***. *cname* specifies the column by name ***cn***.

*type **dtype***
Specifies the data type. Legal data types are I, E, D, and K.

*elem **[list]** | elem **"string"***
Specifies the data elements to be inserted. Numerical data must be enclosed in brackets. Strings must be enclosed in double quotes.

DELETE - Delete contents of cell

dena deletes the contents of a cell specified by line number and column number or column name.

Synopsis

*delete (ch **ichan**) (line **iline**) (cnum **icol** | cname **cn**)*

Description

*ch **ichan***
Specifies the channel number ***ichan***.

*line **lnumber***
Specifies the line number ***lnumber***.

*cnum **ic** | cname **cn***
Specifies the column. *cnum* specifies the column by number ***ic***. *cname* specifies the column by name ***cn***.

Utility Commands

Utility commands are general Monitor commands that may be issued from any Monitor subsystem.

DIR - Print directory of data base file

directory or *dir* displays the data set names and attributes of the contents of a data base file. Note that the first 20 characters of the data set names are printed. To get data set names longer than 20 characters printed, specify *full* (see below).

Synopsis

*directory **setname** (-u **iu**) (short | full | dc)*

Parameters

setname
Specifies the data set name of the data set to be listed. The data set name must be delimited by blanks or by double quotes, and it may not contain blank characters. “Wild-cards” in the data set name specification are permitted according to the rules described earlier in this Chapter.

short
Lis*ts a reduced table of contents.*

full
Lists the full table of contents of the data base file (default).

dc
Lists the full table of contents of the Dynamic Common.

Data set name attributes

*-u **iu***
-u specifies the logical file unit ***iu*** of the data set. ***iu*** must be in the range of *0* to *11*. The unit *0* is assigned to AM data sets only, i.e. data sets residing in the AM buffer when in AM mode (see *set mode* command). If ***iu*** is not specified, the default unit will be selected (as defined by the *unit* command, or by the latest *open* command).

Example

Print directory of all sets ***DISP*** on data base file of unit 3:

```
dir DISP* -u 3
```

Execute script

Scripts are collections of statements placed in a text file which resides in the current directory. Procedures are invoked by name:

@proc_name (p1 p2 ...)

The procedures parameters pi are optional (1 to 9 parameters). Parameters within the procedures must be referenced by $1 $2 etc.

Since the command language does not allow for local variables, all variables within the procedures are global.

Example: Execute a procedure file `proc` which opens a data base file whose name is passed as a parameter to the procedure:

```
# Open file
open unit 1 file $1
```

The script `proc` is invoked by typing

```
@proc demo.db
```

upon which the commands contained in `proc` are interpreted, i.e. the data base file `demo.db` is opened.

HELP - On-line documentation

help displays on-line explanation of the commands. The *Mosaic HTML* viewer is popped up when calling *help*.

Synopsis

help

SYSTEM - Shell command

system issues commands to the operating system (shell). Any valid operating system commands can be started from *system*. This command is particularly useful if command language scripts have to be edited during an interactive session (see example below).

Synopsis

sys ***"commands"***

Parameters

commands
Operating system commands to be started. The commands must be enclosed within double quotes or any other character not contained in the command string.

Example

Edit a script by creating a file *init* and execute it within the current module.

```
syst ’vi init’
...
vi appears, type text and quit vi
...
@init
```

SET - Set Monitor parameters

set modifies various *Monitor* parameters and sets *Monitor* switches. Parameters and switches which have been modified by the *set* command remain active until a new *set* command is issued or until the *Monitor* is left.

Notice: Only one parameter may be selected during each *set* command.

Synopsis

set parameters

Parameters

output ***fname***
Re-directs the output to the file ***fname***. Output is appended to the file ***fname*** if it already exists. To restore the default (terminal) output, set ***fname*** to ***default***.

echo
Enables the echo file output (default). If set, all input typed by the user is copied to the default echo file ***monitor.eco***.

noecho
Disables the echo file output.

echofile ***fname***
Specifies an echo file name.

format ***dtype "format"***
Changes the output format of the data type ***dtype*** (formatted output only). The data types ***dtype*** may be I, E, F, D, Z, X, and A. The format specifier ***format*** is specified according to the *Fortran77* format rules.
Default values for data type E is ***(6e12.4)***, for F and D ***(6d12.4)***, for I ***(10i6)***, and for A ***(20a4)***.

type ***dtype***
Specifies the default data set type which is selected if the ***-t*** attribute is omitted. ***dtype*** may be one of ***A, C, D, E, I, O, P, Z, $***. The default data type is I.

SHOW - Display Monitor parameters

show displays information on various *Monitor* and data base file parameters.

Synopsis

show (files | *formats* | *header* ***(iu)*** | *param* | *size)*

Parameters

files
Displays the open data base file names.

formats
Displays the default print formats (see set format command).

header ***(iu)***
Displays the file header of unit ***iu***.

param
Displays *Monitor* parameters.

size
Displays the file size (in bytes).

VERSION - Display version of MemCom

version displays the current version of *MemCom* and the version of *MemCom* at the time of creation of all open data base file.

Synopsis

version

TIME - Display CPU time

time displays the CPU time of the current *Monitor* run.

Synopsis

time

WAIT - Stop execution

wait stops the execution of the current module and resumes after ***nsec*** seconds.

Synopsis

wait ***nsec***

Description

nsec
Number of seconds to wait until the process continues.

3 Monitor User Guide

This Chapter explains basic features of the *MemCom Monitor* giving some advice on how to use the Monitor.

General Information

File names

MemCom data base file names must be defined according to the rules which pertain to the current operating system. File names may not contain blanks or special characters, i.e. ASCII characters 0 to 31. Examples of file names (UNIX examples):

```
demo.db
/tmp/demo.db
$BIGFILE/demo.db
```

Data set names

A data set name (label) consists of a string separated by blanks. The data set name is limited to 40 characters. Example of data set names:

```
COOR.1
ADIR
DISP.1...*
```

Data set name qualifiers

Qualifiers specify the data set attributes such as the size of a data set, the data set type, or the data set sub-division index. Example:

```
COOR.1 -s 15
```

selects the sub-division index 15 of data set `COOR.1`. *MemCom* requires data sets to be typed for certain operations, such as symbolic data manipulation. The type specification within the *MemCom* Monitor occurs through the `-t` qualifier or through the implicit data set type declaration specified by the `set type` command. Appendix B contains a summary of the data set types recognized by *MemCom*.

Data set descriptors

All data set on the data base files are preceded by a descriptor record. The descriptor record may be used to describe the nature and contents of the data set. The descriptor is transparent to the user, i.e. it is visible only by means of the relational table access commands of the *table* sub-system. By default the descriptor is empty.

Data set descriptor tables should mainly be used to handle relatively small volumes of data. Large volumes of data are handled more efficiently in numerical form. Refer to the *MemCom* Reference Manual for a more detailed description of relational tables and data set descriptors.

The Data Base Manager

The Monitor Data Base Manager system controls direct and indirect access to *MemCom* data sets supporting subsystems such as the *table* subsystem or the `net` subsystem.

The example below opens the data base file `demo.db` on unit 1and prints the data set `MODE.1...1:`

```
open file demo.db unit 1
print MODE.1...1
```

To get a printout of the directory of data base file on unit 1, type

```
dir
```

The tabulated output is listed below:

Set #	SET-NAME	FILE-ADDRS	NWORDS	TYPE	NOF-DIVS	LEN-DIV
1	EPAR	257	480	I	12	40
2	ENAM	993	160	A	40	4
3	TITLE	1409	12	K	0	0
4	MDES.1	1709	512	$	0	0
5	ETAB.1	2477	375040	$	2930	128
6	COOR.1	377773	12213	D	0	0
8	ADIR	402735	100	I	0	0
30	MODE.1...1	415087	24426	E	4071	6
35	MODE.1...6	538497	24426	E	4071	6

To modify the content of a set, make use of the input command. Example: Change the first 3 elements of subset 23 of data set `COOR.1`:

```
input COOR.1 -t F -s 23 [ 1.1 2.2 3.3 ]
```

Relational Tables

Relational tables and data set descriptors may be accessed interactively within the *table* subsystem. The table mechanism is explained in detail in the *MemCom* Reference manual, Chapter 4.

The *table* subsystem is called from within the *Monitor* by typing

```
table
```

To exit from the `table` subsystem, type `end`. Before you can access relational tables and data set descriptors, you must load the relational table from the data base file. Example: Load the relational table `MDES.1`

```
gettab MDES.1
```

`gettab` loads the (existing) relational table `MDES.1` into the table buffer. The command

```
getdes MODE.1...1
```

loads the data set descriptor of the (existing) data set `MODE.1...1` into the table buffer.

A relational table currently in the table buffer is copied back to the data base file by the `puttab` command (relational table) or by the `putdes` command (data set descriptor). Any existing table on the data base file is then overwritten.

A new relational table or descriptor is initialized by the `clear` command:

```
clear
```

All information stored in the table buffer is then destroyed. Note that any new relational table created within `table` subsystem is truncated when it is copied to the data base file. If this table is increased in size (by adding items) it will no more fit into the table already residing on the data base file. You should therefore create the table on the data base file by means of the `reserve` command:

```
reserve -t $ COLORTABLE
```

The command creates a relational table on the data base file. It does not load the table in the table buffer. To load the new table in the table buffer, issue `gettab` after `reserve`. Once a table has been loaded, data objects may be inserted, extracted and modified. To insert a scalar integer, type

```
insert green -t I 23
```

To insert a string, type

```
insert COLOR -t K "green"
```

Example: Insert an integer object `COLORS` containing 5 values:

```
insert COLORS -t I [1 2 3 4 5]
```

Note that the length of the list is defined by the number of values found in the list.

Data may be extracted and printed from relational tables by the `extract` command:

```
extract COLORS -t K
```

The extracted data is either printed on the terminal or redirected to the current alternate output unit. Lists (arrays) are printed by specifying the size (-*l* qualifier).

To replace objects issue the `replace` command. `replace` actually deletes the current item in the table and creates a new entry. It is currently not possible to replace elements of lists.

Objects are deleted by the `delete` command:

```
delete COLORS
```

`delete` automatically compresses the table.

All objects of a relational table are printed on the terminal by the `print` command.

A Monitor Start-Up

The basic versions of the *MemCom Monitor* is operational on the following platforms:

- Various UNIX systems
- VAX/VMS system

Check the release notes for currently supported version numbers of the platform operating system. The *Monitor* might not be supported longer on older versions of operating systems. Note that some options of the *Monitor* may not be operative in a particular implementation.

The *Monitor* on-line documentation has been integrated in the *Mosaic* system. To view the *Monitor* on-line documentation, the *Mosaic* viewer (or any other *HTML* viewer), must be available. The *MemCom* installation procedure tries to find the *Mosaic* or the *Netscape* viewer. If the search is successful, the mon start-up procedure is configured accordingly (see the *-v* parameter of the *Monitor* start-up command line).

The *Monitor* is invoked by starting the *mon* script found in the *bin* subdirectory of the *MemCom* distribution directory.

Synopsis (UNIX operating system)

mon (arguments)

Command line arguments

-dcs ***len***
Sets the Dynamic Common (DC) work space size to ***len*** words. The DC work space size is activated only when in DC mode.
Default: Installation-dependent, usually 100000.

-echo | *-noecho* | *-efile* ***fname***
-echo activates the echo file, i.e. the file which collects all issued *Monitor* scripting language commands. *-noecho* inhibits output to the echo file. *-efile* opens the echo file ***fname***. The default echo file name is ***monitor.eco***.

-l ***lfile***
Invokes the login file ***lfile***. The login file consists of a valid *Monitor* script file.

-m ***manpath***
Sets the Monitor on-line documentation path. Unless the on-line documentation has been moved to another place, *-m* should not be specified.
Default: As set by the *MemCom* installation procedure.

-pX ***value***
Sets the login procedure parameter X to ***value***. Parameters 1 to 4 may be addressed.

-wss ***len***
Sets the *Monitor* work space size to ***len*** words.
Default: Installation-dependent, usually 500000 words.

-v ***htmlvn***
Sets the *HTML* document viewer program name to ***htmlv***n.
Default: ***mosaic***.

Examples

1. Invoke the *Monitor* and execute the login file `init`.

```
mon -l init
```

2. Invoke the *Monitor* and executes the login file `init`. It refers to parameter `$1` which is substituted by the value of *p1*, i.e. `demo.db`.

```
mon -l init -p1 demo.db
```

3. Invoke the *Monitor* by reserving a work space of 5000000 words.

```
mon -wss 5000000
```

B Data Set Types

Any *MemCom* data set has a unique data type identifier. Standard data set types are described in the table below. *MemCom* will also accept and preserve other type specifications than the ones mentioned below. For the non-standard data types, however, there is no built-in relation between logical and physical word counts.

Type	Description
A	Alphanumeric (text). Old-style data representation for text strings.
(blank)	Unspecified format.
C	Complex data of type E (real).
D	Double-precision floating-point format (*).
E	Real floating-point format (*)
F	Floating-point format (real*8 data).
I	Integer format (**)
K	Character string.
P	Text file (extended character string).
S	Extension file.
T	User-defined table.
$	Relational table.
+	Relational data set,
-	System data.
!	Array table.
	* Real and double floating point formats are defined relatively to the machine precision expressed in words, where the word size is expressed in bits or in bytes. As a general rule 32 bit word implementations of MemCom result in 32 bit reals (E) and 64 bit doubles (D) and 64 bit word implementations of MemCom result in 64 bit reals (E) and 128 bit doubles (D). Note that the F format always defines real*8 (f77) or double (C) data formats.
	** The integer format is defined relatively to the machine precision expressed in words, where the word size is expressed in bits or in bytes. As a general rule 32 bit word implementations of MemCom result in 32 bit integers and 64 bit word implementations of MemCom result in 64 bit integers.

Index